```
# === INSTALL DEPENDENSI ===
!pip install -q ultralytics
!pip install -q git+https://github.com/sunsmarterjie/yolov12.git roboflow
supervision flash-attn
!pip install -q geopandas rasterio shapely

# === IMPORT PUSTAKA ===
import os
import cv2
import numpy as np
import rasterio
from rasterio.windows import Window
from rasterio.transform import xy
from shapely.geometry import box as shapely_box
import geopandas as gpd
from ultralytics import YOLO
import supervision as sv
from tqdm import tqdm
import ultralytics
ultralytics.checks()

# === INISIALISASI MODEL YOLO ===
model = YOLO('path/to/best.pt')  # Ubah ke path model YOLO Anda

# === DEFINISI PIPELINE UTAMA ===
def pipeline_yolo_object_detection(input_tif_path, tile_size):
    # Buat folder output berdasarkan ukuran tile
    root_output = f'path/to/output/{tile_size}'
    folders = {
        'tif_tiles': os.path.join(root_output, 'tif_tiles'),
        'jpg_tiles': os.path.join(root_output, 'jpg_tiles'),
        'jpg_bbox': os.path.join(root_output, 'jpg_bbox'),
        'txt_bbox': os.path.join(root_output, 'txt_bbox'),
        'geojson_bbox': os.path.join(root_output, 'geojson_bbox')
    }
    for path in folders.values():
        os.makedirs(path, exist_ok=True)

    # Fungsi untuk memotong citra .tif besar menjadi tile
    def tile_large_tif():
        with rasterio.open(input_tif_path) as src:
            meta = src.meta.copy()
            cols, rows = src.width, src.height
            for i in range(0, rows, tile_size):
                for j in range(0, cols, tile_size):
                    win = Window(j, i, tile_size, tile_size)
                    transform = src.window_transform(win)
                    data = src.read(window=win)
                    if data.shape[1] == 0 or data.shape[2] == 0:
                        continue
```

```
                    tile_name =
f"{os.path.splitext(os.path.basename(input_tif_path))[0]}_{i}_{j}.tif"
                    out_meta = meta.copy()
                    out_meta.update({'height': data.shape[1], 'width':
data.shape[2], 'transform': transform})
                    out_path = os.path.join(folders['tif_tiles'], tile_name)
                    with rasterio.open(out_path, 'w', **out_meta) as dst:
                        dst.write(data)
                    yield out_path, transform, src.crs

    # Konversi tile .tif ke .jpg
    def convert_tile_to_jpg(tile_path):
        with rasterio.open(tile_path) as src:
            if src.count >= 3:
                bands = src.read([1, 2, 3])
            else:
                gray = src.read(1)
                bands = np.stack([gray] * 3)
            img = np.transpose(bands, (1, 2, 0))
            bgr = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
            jpg_name = os.path.splitext(os.path.basename(tile_path))[0] + '.jpg'
            out_jpg = os.path.join(folders['jpg_tiles'], jpg_name)
            cv2.imwrite(out_jpg, bgr)
        return out_jpg, img

    # Konversi koordinat piksel ke koordinat geografis
    def pixel_to_geo(x, y, transform):
        lon, lat = xy(transform, y, x, offset='center')
        return lon, lat

    # Deteksi objek dengan YOLO
    def detect_with_yolo(jpg_path):
        results = model.predict(source=jpg_path, conf=0.25, verbose=False,
imgsz=tile_size)
        detections = sv.Detections.from_ultralytics(results[0]).with_nms()
        return detections

    # Simpan hasil deteksi ke dalam file .jpg, .txt, dan .geojson
    def save_bounding_boxes(jpg_path, detections, transform, crs):
        base = os.path.splitext(os.path.basename(jpg_path))[0]
        image = cv2.imread(jpg_path)
        geo_boxes = []
        txt_lines = []

        labels = [
            f"{model.model.names[class_id].lower()} {conf:.2f}"
            for class_id, conf in zip(detections.class_id, detections.confidence)
        ]

        for box, class_id, conf in zip(detections.xyxy.astype(int),
```

```
detections.class_id, detections.confidence):
            x_min, y_min, x_max, y_max = map(int, box)
            label_name = model.model.names[int(class_id)].lower()
            txt_lines.append(f"{label_name} {conf:.4f} {x_min} {y_min} {x_max}
{y_max}")

            lon1, lat1 = pixel_to_geo(x_min, y_min, transform)
            lon2, lat2 = pixel_to_geo(x_max, y_max, transform)
            geo_boxes.append(shapely_box(lon1, lat1, lon2, lat2))

            cv2.rectangle(image, (x_min, y_min), (x_max, y_max), (0, 255, 0), 2)

        if len(detections.xyxy):
            image = sv.BoxAnnotator().annotate(scene=image, detections=detections)
            image = sv.LabelAnnotator().annotate(scene=image,
detections=detections, labels=labels)

            out_jpg = os.path.join(folders['jpg_bbox'], os.path.basename(jpg_path))
            cv2.imwrite(out_jpg, image)

            txt_path = os.path.join(folders['txt_bbox'], base + '.txt')
            with open(txt_path, 'w') as f:
                f.write('\n'.join(txt_lines))

            gdf = gpd.GeoDataFrame({
                'class': [model.model.names[int(c)].lower() for c in
detections.class_id],
                'confidence': [float(c) for c in detections.confidence],
                'geometry': geo_boxes
            }, crs=crs)
            gdf.to_file(os.path.join(folders['geojson_bbox'], base + '.geojson'),
driver='GeoJSON')

    # === EKSEKUSI PIPELINE UNTUK SETIAP TILE ===
    for tile_path, transform, crs in tqdm(tile_large_tif(), desc=f"Processing tile
size {tile_size}"):
        jpg_path, _ = convert_tile_to_jpg(tile_path)
        detections = detect_with_yolo(jpg_path)
        if len(detections.xyxy):
            save_bounding_boxes(jpg_path, detections, transform, crs)

# === JALANKAN PIPELINE UNTUK BERBAGAI TILE SIZE ===
input_tif_path = 'path/to/orthophoto.tif'  # Ganti dengan path citra orthophoto
.tif

for size in [320, 640, 1280, 1920, 2560]:
    pipeline_yolo_object_detection(input_tif_path, tile_size=size)
```